

**APPLICATION
FOR
UNITED STATES LETTERS PATENT**

APPLICANT NAME: Masek et al.

TITLE: METHOD, SYSTEM AND PROGRAM PRODUCT FOR
TESTING A SERVER APPLICATION USING A
REENTRANT TEST APPLICATION

DOCKET NO.: LOT920030024US1

INTERNATIONAL BUSINESS MACHINES CORPORATION

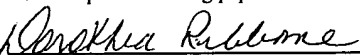
CERTIFICATE OF MAILING UNDER 37 CFR 1.10

I hereby certify that, on the date shown below, this correspondence is being deposited with the United States Postal Service in an envelope addressed to the Mail Stop Patent Application, Commissioner for Patents, Alexandria, VA 22313-1450 as "Express Mail Post Office to Addressee" Mailing Label No. EV225574924US

on September 25, 2003

Dorothea Rubbone

Name of person mailing paper


Signature

September 25, 2003
Date

METHOD, SYSTEM AND PROGRAM PRODUCT FOR TESTING A SERVER APPLICATION USING A REENTRANT TEST APPLICATION

Background of the Invention

1. Field of the Invention

[0001] The present invention generally relates to a method, system and program product for testing a server application using a reentrant test application. Specifically, the present invention provides an efficient and effective way to test an application running on a server by allowing multiple instances of a test application to be instantiated within a single process on a test machine.

2. Related Art

[0002] As the use of computer hardware and software becomes more prevalent, the need for providing improved product testing is increasing. This is especially the case as more applications are implemented in a network environment. More and more applications on clients are connecting directly to server applications using complex and proprietary protocols. Specifically, before a server application is released, it would be advantageous to first test it in a realistic environment. For a server application, proper testing would be to simulate multiple clients or users running the external network application that uses the application as it runs on a server. To do so, a suitable load must be driven against the server.

[0003] Heretofore, many approaches have attempted to provide product testing. For example, “record and playback” products such as LOAD RUNNER, RATIONAL ROBOT or APACHE JAKARTA, record actual application traffic and then play it back to the application. Although this approach could be marginally effective for simple traffic flows, much application traffic is more complex. If the application uses a dynamic protocol then the playback tool must parse the network traffic and change the dynamic values to create a valid load. The record/playback tool must implement record and playback modules for each protocol. Another approach spawns a process such as a Java virtual machine (JVM) per test user. However, given the expense in providing multiple processes, this approach does not scale very well. Still yet, another approach is known as the “test bean” approach and involves running application protocol interfaces (APIs) against the server. This approach, however, does not reflect how the application truly functions on the client. As such, realistic user operation is not provided. In addition if the API’s do not satisfy the reentrancy requirement then the applications on the test drivers will not act like a single user on the server.

[0004] In view of the foregoing, there exists a need for a method, system and program product for testing a server application using a reentrant test application. To this extent, a need exists for a system that can provide realistic simulation of a user operating the application. Further, a need exists for a system that can test the application efficiently.

Summary of the Invention

[0005] In general, the present invention provides a method, system and program product for testing a server application using a reentrant test application. Specifically, under the present invention, the APIs of the test application are identified and a test script that is capable of invoking the identified APIs is provided in a test application. If the test application satisfies reentrancy requirements, the test script is executed, which instantiates a plurality of instances of the test application using threads. Specifically, each instance of the test application runs in a separate thread. Because the test application is reentrant, the plurality of instances can run within a single process (e.g., JVM) with each thread generating the load of a separate user on the server on which the application is loaded. This provides a realistic simulation of use of the server application by multiple users running the test application.

[0006] A first aspect of the present invention provides a method for testing a server application using a reentrant test application, comprising: providing a test application that satisfies reentrancy requirements on a client; and instantiating a plurality of instances of the test application using threads, wherein each of the plurality of instances of the test application run within a single process.

[0007] A second aspect of the present invention provides a system for testing a server application using a reentrant test application, comprising an application instantiation system for instantiating a plurality of instances of the test application on a client using threads if the test application satisfies reentrancy requirements, wherein each of the plurality of instances of the test application run within a single process.

[0008] A third aspect of the present invention provides a program product stored on a recordable medium for testing a server application using a reentrant test application, which when executed, comprises: program code for instantiating a plurality of instances of a test application on a client using threads if the test application satisfies reentrancy requirements, wherein each of the plurality of instances of the test application run within a single process.

[0009] Therefore, the present invention provides a method, system and program product for testing a server application using a reentrant test application.

Brief Description of the Drawings

[0010] These and other features of this invention will be more readily understood from the following detailed description of the various aspects of the invention taken in conjunction with the accompanying drawings in which:

[0011] Fig. 1 depicts a system for testing a server application using a reentrant test application according to the present invention.

[0012] Fig. 2 depicts a conceptual diagram showing the operation of the system of Fig. 1.

[0013] Fig. 3 depicts a conceptual diagram showing the operation of a prior art system for testing a server application,

[0014] Fig. 4 depicts a method flow diagram, according to the present invention.

[0015] The drawings are merely schematic representations, not intended to portray specific parameters of the invention. The drawings are intended to depict only typical

embodiments of the invention, and therefore should not be considered as limiting the scope of the invention. In the drawings, like numbering represents like elements.

Detailed Description of the Invention

[0016] As indicated above, the present invention provides a method, system and program product for testing a server application using a reentrant test application. Specifically, under the present invention, the APIs of the test application are identified and a test script that is capable of invoking the identified APIs is provided in a test application. If the test application satisfies reentrancy requirements, the test script is executed, which instantiates a plurality of instances of the test application using threads. Specifically, each instance of the test application runs in a separate thread. Because the test application is reentrant, the plurality of instances can run within a single process (e.g., JVM) with each thread generating the load of a separate user on the server on which the application is loaded. This provides a realistic simulation of use of the server application by multiple users running the test application.

[0017] It should be appreciated in advance that the term “server application” is intended to mean any type of computer application program that can be loaded on a server. For example, the server application could be a network application (e.g., a same time meeting application) that runs on a server, and with which users interact through clients. To this extent, the application can be a JAVA-based or other type of application. In any event, the test application on a client that is driving load against the server application should satisfy reentrancy requirements. Specifically, as will be further described below, the

present invention will instantiate multiple instances of the test application to test the server application. As such, multiple test applications will be running in the same process, sharing the same services and memory space. Reentrancy prevents the test applications from interfering with each other.

[0018] Referring to Fig. 1, a system 10 for testing a server application 34 loaded on a server 14 is shown. Under the present invention, testing program 36 on test client 12 (also referred to as a “driver”) instantiates a plurality of instances of test application 43 using threads so that all of the instances can run within a single process 32 (e.g., JVM). This simulates load from a plurality of users in an efficient manner. To this extent, the teachings of the present invention are typically implemented to simulate a load that would normally occur in a network environment such as the Internet, a wide area network (WAN), a local area network (LAN), a virtual private network (VPN), etc. As known, communication between actual clients and server 14 could occur via a direct hardwired connection (e.g., serial port), or via an addressable connection that may utilize any combination of wireline and/or wireless transmission methods. Server 14 and the clients may utilize conventional network connectivity, such as Token Ring, Ethernet, WiFi or other conventional communications standards. Moreover, connectivity could be provided by conventional TCP/IP sockets-based protocol. In this instance, the clients could utilize an Internet service provider to establish connectivity to server 14.

[0019] In any event, as depicted, server 14 generally comprises central processing unit (CPU) 20, memory 22, bus 24, input/output (I/O) interfaces 26 and external devices/resources 28. CPU 20 may comprise a single processing unit, or be distributed

across one or more processing units in one or more locations, e.g., on a client and computer system. Memory 22 may comprise any known type of data storage and/or transmission media, including magnetic media, optical media, random access memory (RAM), read-only memory (ROM), a data cache, etc. Moreover, similar to CPU 20, memory 22 may reside at a single physical location, comprising one or more types of data storage, or be distributed across a plurality of physical systems in various forms.

[0020] I/O interfaces 26 may comprise any system for exchanging information to/from an external source. External devices/resources 28 may comprise any known type of external device, including speakers, a CRT, LCD screen, hand-held device, keyboard, mouse, voice recognition system, speech output system, printer, monitor/display, facsimile, pager, etc. Bus 24 provides a communication link between each of the components in server 14 and likewise may comprise any known type of transmission link, including electrical, optical, wireless, etc.

[0021] Although not shown, additional components, such as cache memory, communication systems, system software, etc., may be incorporated into server 14.

Further, it is understood that test client 12 will typically include the same components (e.g., CPU, memory, etc.) as shown and described with reference to server 14. These components have not been separately shown and discussed for brevity.

[0022] Shown within process 16 in memory 22 of server 14 is application 34 being tested. In addition, shown within process 32 of test client 12 is testing program 36. As indicated above, and assuming test application 43 is reentrant, testing program 36 will instantiate a plurality of instances of test application 43 using threads within the single

process 32. This provides a efficient, cost effective and realistic manner of testing application 34. As shown, testing program 36 includes interface identification system 38, application instantiation system 40 and data collection system 42. Under the present invention, interface identification system 38 will first identify the APIs of test application 43. Once the APIs are identified, a test script that can invoke the APIs can be provided by administrator 18 to the test client 12 or the like and stored in storage unit 30 connected to the client 12. Storage unit 30 can be any system (e.g., database) capable of providing storage for information under the present invention. As such, storage unit 30 could include one or more storage devices, such as a magnetic disk drive or an optical disk drive. In another embodiment, storage unit 30 includes data distributed across, for example, a local area network (LAN), wide area network (WAN) or a storage area network (SAN) (not shown).

[0023] Assuming, the reentry requirements are satisfied, application instantiation system 40 will run the test script to call the APIs and instantiate a plurality of instances of test application 43 using threads. Because test application 43 is reentrant, all of the instances can run with within a single process 32 without interfering with each other. Specifically, each instance of test application 43 that is instantiated corresponds to a separate thread. Each thread in turn is associated with a different connection to server 14. Referring to Fig. 2, this concept is shown in greater detail. Specifically, Fig. 2 depicts a conceptual diagram of the present invention. As depicted, test client 12 will execute the test script, which will cause instances of test application 43A-C to be instantiated. Each instance 43A-C uses a separate thread and runs within process 32. Accordingly, if testing program

36 is attempting to simulate use by 100 users, 100 instances of test application 43 will be instantiated within process 32 using 100 threads and 100 connections to server 14.

[0024] This is in stark contrast to previous methods in which multiple processes were required to instantiate multiple instances of an application. For example, referring to Fig. 3, the concept of the previous method is shown. As depicted, for each instance of application 52A-B that is instantiated by test client 56, a separate process 58A-B and instance of testing program 60A-B must be provided on test client 56. Since each process 58A-B can consume 10-20 megabytes of memory, the method shown in Fig. 3 is costly and inefficient.

[0025] Referring back to Fig. 1, as the test is running, data collection system 42 will collect data corresponding thereto. For example data collection system 42 can not only keep track of test specifics such as time and date information, but it can also monitor and track (e.g., in storage unit 30) the quantity and identity of the instances instantiated. This tracking can also include how application 34 performed under test. Administrator 18 can view the results and make any necessary changes to application 34.

[0026] Referring now to Fig. 4 a method flow diagram 100 according to the present invention is shown. As depicted, first step S1 is to provide a test application that meets reentrancy requirements on a client. Second step S2 is to identify APIs associated with the test application. Third step S3 is to provide a test script for invoking the APIs. Fourth step S4 is to instantiate a plurality of instances of the test application using threads. This allows the plurality of instances to effectively run within a single process.

[0027] It should be understood that the present invention can be realized in hardware, software, or a combination of hardware and software. Any kind of computer system(s) - or other apparatus adapted for carrying out the methods described herein - is suited. A typical combination of hardware and software could be a general purpose computer system with a computer program that, when loaded and executed, carries out the respective methods described herein. Alternatively, a specific use computer, containing specialized hardware for carrying out one or more of the functional tasks of the invention, could be utilized. The present invention can also be embedded in a computer program product, which comprises all the respective features enabling the implementation of the methods described herein, and which - when loaded in a computer system - is able to carry out these methods. Computer program, software program, program, or software, in the present context mean any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following: (a) conversion to another language, code or notation; and/or (b) reproduction in a different material form.

[0028] The foregoing description of the preferred embodiments of this invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and obviously, many modifications and variations are possible. Such modifications and variations that may be apparent to a person skilled in the art are intended to be included within the scope of this invention as defined by the accompanying claims.